Runtime Decision Making Under Uncertainty in Autonomous Vehicles

Vibhu Gautam, Youcef Gheraibia, Rob Alexander, and Richard Hawkins
Department of Computer Science, University of York, UK
{vibhu.gautam,youcef.gheraibia,rob.alexander,richard.hawkins}@york.ac.uk

Abstract

Autonomous vehicles (AV) have the potential of not only increasing the safety, comfort and fuel efficiency in a vehicle but also utilising the road bandwidth more efficiently. This, however, will require us to build an AV control software, capable of coping with multiple sources of uncertainty that are either preexisting or introduced as a result of processing. Such uncertainty can come from many sources like a local or a distant source, for example, the uncertainty about the actual observation of the sensors of the AV or the uncertainty in the environment scenario communicated by peer vehicles respectively. For AV to function safely, this uncertainty needs to be taken into account during the decision making process. In this paper, we provide a generalised method for making safe decisions by estimating and integrating the Model and the Data uncertainties.

1 Introduction

In an AV's software pipeline, the uncertainty arising from various sources is critical for safe decision making. Due to the recent advancement in Machine Learning (ML) techniques, especially Neural Networks (NN), the software pipeline of an AV is heavily dependent on data related to its environment and this data comes from the sensors. The key data sources in the software pipeline of an AV are the LIDAR, RADAR, GPS, camera, etc. As these sources are prone to measurement fluctuations, there is always some uncertainty or noise in the data which they provide, for example, uncertainty due to variation in sensor resolution, internal sensor noise, measurement fluctuations caused by changes in the weather like rain, dust, etc. This gives rise to uncertainty about how the sensor data corresponds to the ground truth. Although recent advancements in sensor technology have greatly reduced such inaccuracies, they still remain of significant concern (Schwarting, Alonso-Mora, and Rus 2018) (McAllister et al. 2017).

In the software pipeline of a typical AV, the perception task is heavily dependent on data and advanced ML model techniques, both of which are prone to uncertainty. This uncertainty can lead to incorrect predictions and therefore, jeopardize the safety of the AV. Hence, for the safety of an AV, it becomes imperative that we incorporate these uncertainties in the decision making process. (Macfarlane and Stroila 2016)

One recent ML technique, known as the Convolutional Neural Network (CNN), has been widely adopted across both the industry and the research, primarily because of its at par human level accuracy in dealing with various image recognition challenges and for providing robustness to the Data and Model Uncertainty. (CNN are robust to large variation in input data). The perception task of an AV also utilises CNN techniques for various classification and object detection tasks. (Stallkamp et al. 2012)

For a safety critical application like an AV, it becomes imperative that in perception tasks, such CNN models not only have high accuracy but are also able to estimate and utilise the Data and Model Uncertainty for decision making. Recent advances in the area of Probabilistic Convolutional Neural Networks (PCNN) have provided a way to estimate the Data and Model Uncertainty for object classification. (McAllister et al. 2017)

Data Uncertainty arises from sensor noise or measurement fluctuations caused by changes in weather conditions like rain, dust, etc., whereas, Model Uncertainty arises because the ML models learn from data and are not explicitly programmed to perform certain tasks (Kendall and Gal 2017). Like any other ML technique, CNN are also inherently uncertain because the model they have learned is always an imperfect representation of the complex world (Gauerhof, Munk, and Burton 2018).

Bayesian Networks (BN) are an effective technique for decision making under uncertainty, and are utilised heavily for such tasks across domains (Koller and Friedman 2009). However, it has not yet been shown how to use BN to estimate and utilise uncertainties arising specifically from tasks like classification or object detection.

In this paper, we present a method that addresses the challenge of managing the uncertainty from PCNN by using BN for decision-making. Our method links the outputs from a PCNN to a predefined BN. At runtime, the output from the PCNN is used as evidence for nodes of the BN. This allows us to estimate the probability of being in a certain state while taking into account uncertainties arising at runtime. These state probabilities can be used to ensure that safe decisions are taken.

2 Background and related works

The challenges of decision making in an AV, which is safety critical in nature, is that they require robust guarantees to assure safety, security, assurance and other dependable characteristics (Burton, Gauerhof, and Heinzemann 2017) (Gauerhof, Munk, and Burton 2018). Some of the recent work which tries to bridge various decision making techniques with the safety of an AV have shown promising results, for example, Papadoulis et al. (Papadoulis, Quddus, and Imprialou 2019) proposed a runtime decision making control algorithm for AV. The algorithm supported both lateral and longitudinal decision making and was shown to improve road safety by reducing road conflicts. For safer decision making in an AV, Furda et al. (Furda and Vlacic 2011) used Petri net for choosing a safe manoeuvre and Multi Criteria Decision Making (MCDM) model for improving comfort and efficiency under multiple criteria. Katrakazas et al. (Katrakazas, Quddus, and Chen 2019) proposed the usage of Dynamic Bayesian Networks (DBN) to enhance the risk assessment for AV. In order to increase the safety of automated driving, DBN were used to estimate the risk of collision by providing comprehensive reasoning for unsafe driving behaviour.

Though these techniques yield good results, none of these solutions address how to estimate uncertainties arising from perception tasks, or how to take these uncertainties into account during the decision making process.

Work done by (Kabir et al. 2019) tries to utilise uncertainties during runtime in an AV by proposing a conceptual framework for runtime safety analysis using BN and State Machines (SM) in a Platooning Scenario. BN proposed in this architecture are used to address issues of uncertainty in data and to produce runtime probabilistic confidence of being in a certain state. However, the authors do not discuss the methods used for complex tasks like object detection. For example, in their framework, for detecting speed from road signs, they depend on external sources such as roadside infrastructure. It is therefore not clear how various uncertainties can be captured. In our work, we extend this framework to show how these uncertainties can be estimated at runtime and integrated into a BN for safe decision making.

3 Proposed Method

Using (Kabir et al. 2019) work as a reference for our proposed method, at design-time, we model the failure behaviour of the system as a SM. The states of the SM are based on a detailed study of both the environment in which the AV system needs to function and the possible hazards and failures that the AV may encounter. SM have been extensively used to model the failures and faults of a complex system into a chain of simpler states.

Like (Kabir et al. 2019), we use an executable BN, which can be used at runtime to produce the probability of being in a certain state. BN provide a very powerful way to infer the relationship between a large number of random variables which are represented in the form of a Directed Acyclic Graph (DAG). BN also allow us to factor large joint probability distributions by capturing the independence among various random variables.

In (Kabir et al. 2019) framework, any safety failure in the system is defined using a SM and then an executable BN is used to generate the probability of being in certain state. We extend on this framework by proposing a method for estimating both the Data and the Model Uncertainties from the classification task and utilizing them for decision making using the BN. We use PCNN to provide estimates of the Data and the Model Uncertainty along with the Label Prediction for the classification task.

PCNN produces probabilistic understanding of Deep Learning models by inferring the distribution over NN parameters, i.e., Weights and Biases. This distribution over NN parameters allows us to estimate the Model and Data Uncertainty. This estimate of Model and Data Uncertainty are added to get a single value for Total Uncertainty, which is then normalised using logistic regression to present probability of correct classification (Gal and Ghahramani 2015). This probability becomes the runtime evidence for the nodes of the BN. In the next section, we discuss in detail, how PCNN is used to estimate the Model and the Data Uncertainty.

3.1 Estimating Model Uncertainty

Model Uncertainty tells our ignorance about which model parameter best fits the underlying data. In the case of NN, where the model training (learning) process is stochastic in nature, there can be different values for model parameters leading to similar prediction accuracy. Therefore, using PCNN, we can estimate our ignorance regarding which model parameters generated our underlying data (Kendall and Gal 2017). Owing to their large parameter space, estimating Model Uncertainty is a non-trivial task, especially in case of NN (Hinton and van Camp 1993). In addition, as discussed in the previous section, similar to other ML techniques, any NN based technique is also inherently uncertainty. Hence, for safety critical applications, we need methods to estimate this uncertainty and use it for safe decision making.

In a PCNN, exact inference of posterior distributions over a large parameter space, like a Kernel in PCNN, is intractable. Possible methods which exist consist of the Sampling Methods, the Variational Inference Methods or the Ensemble Methods (Graves 2011) (Osband et al. 2016). Sampling Methods and Ensemble methods, both suffer from very high latency in real time usage, for example, when used in an AV. A recent work proposed (Gal and Ghahramani 2015) Random Neuron Dropout during runtime as a method for Approximate Variational Inference. This method only requires dropouts in Forward Passes at runtime. The average stochastic Forward Passes are then interpreted as Bernoulli Approximate Variational Inference. Additionally, to handle any latency issues, PCNN can be deployed for runtime in a distributed manner.

In a given dataset, the input feature space is defined by $\mathrm{X} = [\mathrm{x}_1, .., \mathrm{x}_\mathrm{n}]$, and the output to be predicted is defined as $\mathrm{Y} = [\mathrm{y}_1, .., \mathrm{y}_\mathrm{n}]$. The usage of dropouts at runtime allows us to use the distribution over Weights and Biases which can later be used to calculate the Mean of the Predictive Posterior Distribution (y^*) for any new data (x^*) by taking the Mean of the SoftMax output Score for N number of Forward Passes. Finally, the Model Uncertainty can be captured in the form of Shannon Entropy (SE) (Feng, Rosenbaum, and Dietmayer 2018).

3.2 Estimating Data Uncertainty

Data Uncertainty captures the noise which is inherently present in the sensor data. PCNN help us to quantify the noise in the data as it can be trained to learn this noise in an unsupervised manner. This uncertainty in the data, which is learned by modifying the loss function of PCNN, tells us the noise inherently present in the data (Leung and Bovy 2019) (Kendall and Gal 2017). For classification tasks, in the output layer, in addition to the number of neurons corresponding to the number of classes, an extra neuron is added and the loss function is modified to incorporate for this additional neuron. This allows us to train the extra neuron in an unsupervised manner to learn the uncertainty in the data.

Unlike Model Uncertainty, we do not need to run multiple Forward Passes to capture Data Uncertainty. Also, in case of the latter, uncertainty cannot be reduced using additional data.

3.3 Decision Making using BN

Figure 2, represents the BN for the runtime decision making of the Platooning System. The inference is based on several parameters and inputs, i.e., the distance between the Follower and the Leader, the safe distance, the threshold (proximity in terms of distance), allowed error in distance, the current speed, the speed limit, the validity of speed values, and the detection quality of the Leader and the Follower. The system state is estimated based on the values of the previous parameters and inputs.

For the "Speed Limit" and the "Valid Speed Limit" nodes of the BN, evidence comes for the PCNN. For all other nodes we have generated data artificially. We generated various test case scenario and checked the working of the BN when using the Data and Model Uncertainty from the PCNN. As a simple rule, the state with the highest probability is selected, however, in cases where the probabilities of two or more different states are equal, to avoid a deadlock, the system designer can define a set of rules. In cases where two states have highest and approximately equal probability, safety goals can be ensured by using predefined rules to choose a particular state. For instance, the more safety-critical state can be chosen in the case of a tie.

4 Experiments

In this section, we describe the implementation of our proposed method by using a conceptual platooning case study used by (Kabir et al. 2019). We extend the case study by using PCNN for capturing the Data and Model Uncertainty. We also perform an experiment to test whether or not the safety of our system is ensured when we utilise the uncertainty arising from both the Data and the Modeling tasks.

4.1 Platooning Case Study

The case study we use is a Platooning Scenario consisting of two vehicles, the Follower and the Leader. These vehicles operate in Cooperative Adaptive Cruise Control (CACC), tasked to ensure that a Safe Distance is maintained between the two vehicles. For the Platooning Scenario, the following conditions (Reich 2016) must be ensured and verified at runtime:

– Condition 1: $d \geq d_s$, where d and d_s are the distance between the two vehicles and the minimum safety distance respectively.

– Condition 2: Current Vehicle Speed $\leq$ Speed Limit, where former is the current speed of the vehicles and the latter is the speed limit on the road.

– Condition 3: Any ambiguity arising while checking the validity of the input data, is modeled to ensure the safety of the system and the system utilises only correct input data for decision making.

A SM is used to model the failure behaviour (Machin et al. 2016) of the Platooning System. Based upon the three conditions above, the States and corresponding Actions, to ensure the safety of the system, have been summarised in Table 1 and the SM diagram in Figure 1.

An executable BN can be created to produce the system's probability of being in a certain SM state. The BN model and the PCNN used, as shown in Figure 2, contain both the quantitative and the probabilistic safety parameters for inferring the system's state at runtime. The BN nodes of "Speed", "Speed Limit", "Distance from Follower", "Safe Distance", are all quantitative parameters. These quantitative parameters are used for checking the safety condition related to Speed and Distance, as mentioned in the SM. The "Leader detected by Follower", "Follower detected by Leader" and "Valid Speed Limit", are all probabilistic parameters used for checking the validity of the input data.

The safety of the Platooning Scenario, as defined in the SM in Figure 1, is based on three conditions, i.e., Safe Speed, Safe Distance and Ambiguity. These condition are also represented in the BN. The "Speed Check"

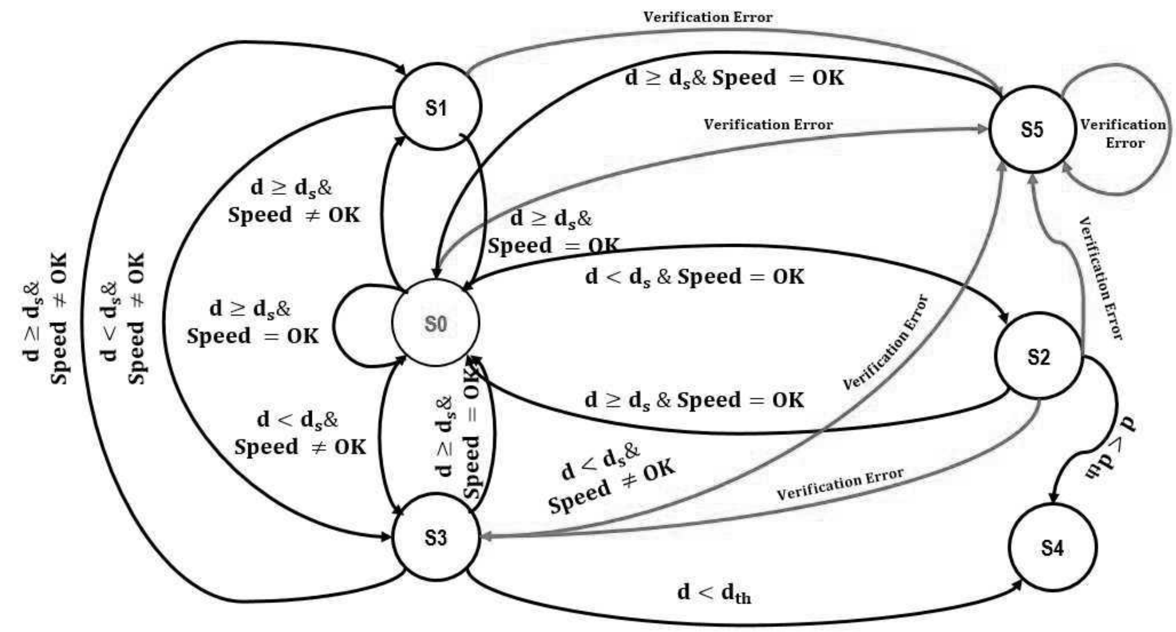

Figure 1: State Machine for the Platooning case study (Kabir et al. 2019)

State	Description	Action
S0	The safety condition of safe distance is fulfilled and the Follower is driving within the speed limit of the road.	The state is safe, therefore, continue driving.
S1	The safety condition of safe distance is fulfilled but the Follower is driving above the speed limit of the road.	Decelerate to fall within the speed limit.
S2	The safety condition of safe distance is not fulfilled and the Follower is driving within the speed limit of the road.	Decelerate to increase the distance with the Leader until safety condition is fulfilled.
S3	The safety condition of safe distance is not fulfilled and the Follower is driving above the speed limit of the road.	Decelerate to achieve a safe distance with the Leader and fall within the speed limit.
S4	The safety condition of safe distance is not fulfilled, the Follower is driving above the speed limit of the road, and is driving too close to the Leader.	Brake to stop driving.
S5	Safety condition of safe distance and/or speed limit cannot be verified.	Switch to ACC mode.

Table 1: Various States and corresponding Actions in the Platooning Scenario (Kabir et al. 2019)

node is responsible for producing probabilistic guarantees of maintaining a safe speed. This is achieved through two child nodes, namely, “Valid Speed Limit”, representing a certificate about the validity of the speed limit and “Speed Within Limit”, monitoring the legality of the vehicle’s current speed by comparing it with the current speed limit. Similarly, “IsSafe” is responsible for producing probabilistic guarantees of maintaining Safe Distance between the Leader and the Follower. The condition for Ambiguity is monitored by the “Detection Quality” node, which provides a guarantee about the detection by both the Leader and the Follower vehicles.
In (Kabir et al. 2019) the validity of the estimate of the speed limit is determined by getting a “certificate” of the speed limit from the external infrastructure. Instead, in our method, we assess the validity of the detected speed limit by using uncertainty estimates from the PCNN.

4.2 Implementation of PCNN

In this section we discuss the implementation of the PCNN for the Platooning Case Study. For the platooning working example, we provide evidence of speed detected from the speed sign board, which is used in the “Speed Limit” node of the BN. Also, the probabilistic confidence of this prediction is used in the “Valid Speed Limit” node of the BN. The PCNN used was trained using a Traffic Sign Dataset where traffic signs were detected from images and uncertainty in the results was quantified using PCNN discussed earlier. The German Traffic Sign Recognition Benchmark Dataset (Stallkamp et al. 2012) (Houben et al. 2013) is a well estab-

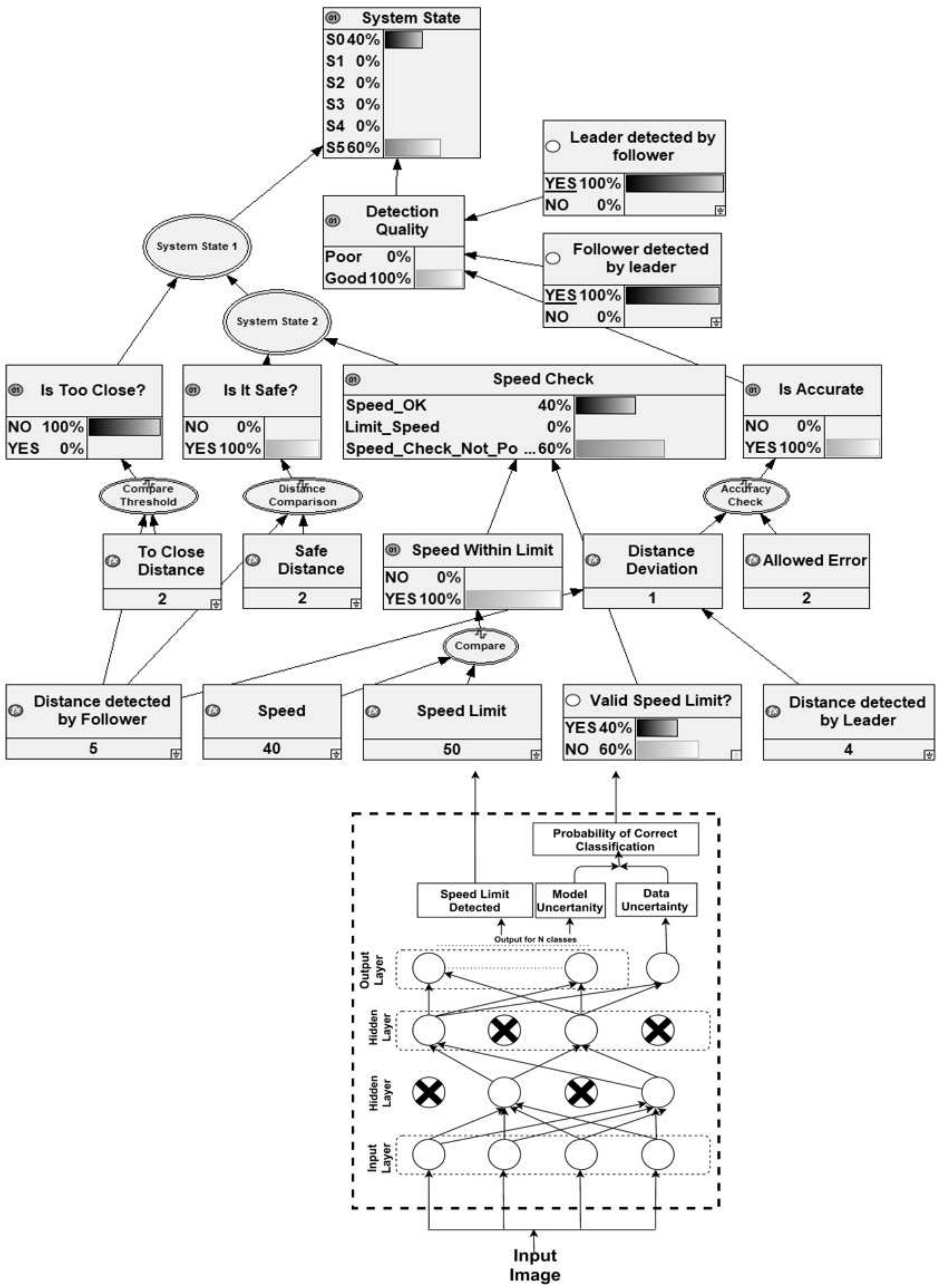

Figure 2: Bayesian Network from (Kabir et al. 2019) framework with PCNN input; Test scenario B3

lished benchmark in the area of automatic traffic sign recognition. This dataset consists of about 50,000 traffic sign images reflecting variations in the visual appearances of signs because of weather conditions, occlusion, rotations, illumination, distance, etc. It consists of 43 classes having unbalanced class frequencies. By default, it is divided into a Training Dataset and a Testing Dataset with 39209 training image and 12630 test-

ing images.
For easy implementation of PCNN, we used AstroNN API, which is built on top of Keras and Tensorflow. For estimating Model Uncertainty, the runtime dropout is implemented by ”MCdropout” layer of the AstroNN API (Leung and Bovy 2019). The dropout rate used was 20 percent. The Data Uncertainty is estimated in the last layer of the architecture as shown in Table 2 and is represented as a ”varianceoutput” layer in the AstroNN API. Speed Sign Detection and the Total Uncertainty in predictions is the output from PCNN and these become the evidence for the nodes of BN, i.e., “Speed limit” and “Validity of speed limit”.

The simple model with 20 epochs was producing a training accuracy of above 95 percent for multiple runs. Also, the test data accuracy was 90 percent and above. Fig 3a) shows how we have high accuracy corresponding to lower value of Total Uncertainty.

The uncertainty measures produced by PCNN are numeric values and not a probability distribution as is required for probabilistic inference in BN. To address this issue, we convert the uncertainty measure, i.e., the sum of the Model and Data Uncertainty, into a probability of correct classification by using a logistic regression and it is implemented with a popular pymc3 library. The results in Figure 3b), show how low uncertainty correlates highly with the probability of correct prediction.

Layer	Output Shape	Parameters
Input Layer	(None,40,40,3)	0
Conv2D	(None,40,40,8)	224
Activation	(None,40,40,8)	0
MCDropout	(None,40,40,8)	0
Conv2D	(None,40,40,16)	1168
Activation	(None,40,40,16)	0
MCDropout	(None,40,40,16)	0
MaxPooling2D	(None,10,10,16)	0
Flatten	(None,1600)	0
Dense	(None,256)	409856
MCDropout	(None,256)	0
Dense	(None,128)	32896
Activation	(None,128)	0
Dense	(None,43)	5547
Dense	(None,43)	5547
varianceoutput(Dense)	(None,43)	5547

Table 2: Probabilistic Neural Network Architecture used in the Experiment

The remaining setup and assumptions for the experiment remain the same as used by (Kabir et al. 2019).

In the next section, we discuss the results we performed to show how our approach can incorporate data and model uncertainties to ensure the overall safety of the Platooning System.

5 Results

To test the working of the proposed method, we generated two Test Scenarios. Scenario A, where the evidence provided to “Validity of speed limit” is considered to be 100% for each test case, and Scenario B, where the the probabilistic uncertainty output from PCNN is used. The results for the tests performed for each scenario are summarised in Table 3 and Table 4.

Parameter	A1	A2	A3	A4
Distance by Follower(m)	5.0	3.0	3.0	2.0
Distance by Leader (m)	5.5	3.5	3.5	2.5
Safe distance (m)	4.0	4.0	4.0	4.0
Too close distance (m)	2.0	2.0	2.0	2.0
Allowed error in distances (m)	2.0	2.0	2.0	2.0
Speed (miles/h)	55	45	55	55
Speed limit (miles/h)	50	50	50	50
Validity of speed limit	100%	100%	100%	100%
Leader detected by follower	100%	100%	100%	100%
Follower detected by leader	100%	100%	100%	100%
State Estimated	S1	S2	S3	S4
	100%	100%	100%	100%

Table 3: Test Cases in Scenario A and corresponding results

Firstly, we discuss the results obtained for test cases in Scenario A, where the evidence provided to “Validity of speed limit” is considered to be 100% for each of the following test case:

– Test Case A1: the “Speed” of the Follower is more than the “Speed Limit” and all other safety conditions are met, therefore, State S1 (Decelerate to fall within the speed limit) is selected with 100% probability.

– Test Case A2: the “Distance detected by Leader” and “Distance detected by Follower” is less than the “Safe distance”, and all other safety conditions are met, therefore, State S2 (Decelerate to increase the distance with the Leader until safety condition is fulfilled) is selected with 100% probability.

– Test Case A3: the “Distance detected by Leader and “Distance detected by Follower” is less than the “Safe distance” and “Speed of the Follower” is more than the “Speed Limit”, therefore, State S3 (Decelerate to achieve a safe distance with the Leader and fall within the speed limit) is selected with 100% probability.

– Test Case A4: the Follower is driving above the “Speed Limit” and is also “Too close” to the Leader therefore, State S4 (Brake to stop driving) is selected with 100% probability.

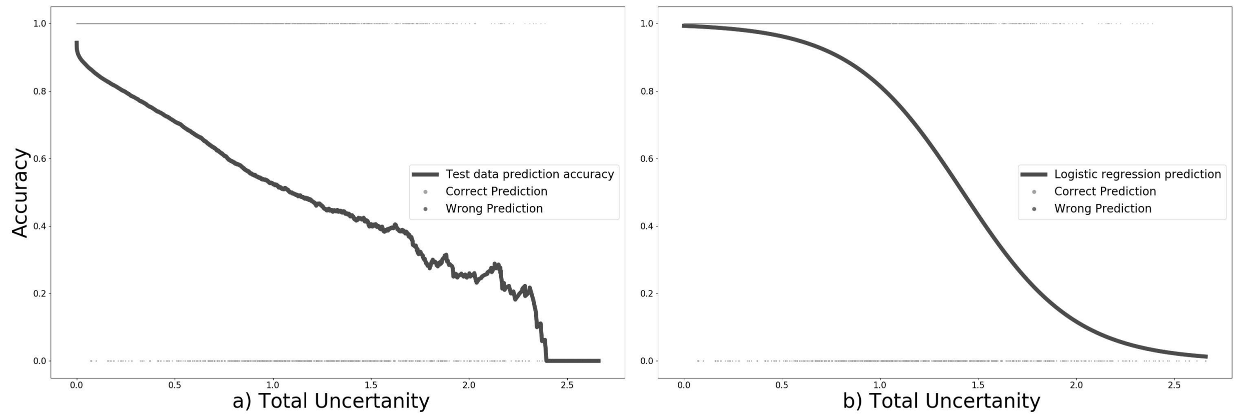

Figure 3: a) The plot shows Total Uncertainty vs Accuracy for the test data, b) The plot shows output of logistic regression as Total Uncertainty vs Probability of Correct Classification

Parameter	**B1**	**B2**	**B3**	**B4**
Distance by Follower(m)	5.0	5.0	5.0	5.0
Distance by Leader (m)	4.0	4.0	4.0	4.0
Safe distance (m)	2.0	2.0	2.0	2.0
Too close distance (m)	2.0	2.0	2.0	2.0
Allowed error in distances (m)	2.0	2.0	2.0	2.0
Speed (miles/h)	40	40	40	40
Speed limit (miles/h)	50	50	50	50
Validity of speed limit	100%	70%	40%	0%
Leader detected by follower	100%	100%	100%	100%
Follower detected by leader	100%	100%	100%	100%
State Estimated	S0	S0	S5	S5
	100%	70%	60%	100%

Table 4: Test Cases in Scenario B and corresponding results

In Scenario B, all the safety conditions, as described in the SM, are met, but instead of always considering the probability of the "Valid Speed Limit" detected as 100%, the probabilistic uncertainty output from PCNN is used. As seen below, the different test cases results in both a change in the probability of the output state and the output state selected:

– Test Case B1: all the safety conditions are met, and there is 100% confidence in the validity of the speed limit, therefore State S0 (The state is safe, therefore, continue driving) is selected with 100% probability.

– Test Case B2: as in Test Case B1, the safety conditions are met and the evidence provided to the nodes in the BN are the same except for the "Valid Speed Limit" node, which gets the normalised input from PCNN. Here, the "Valid Speed Limit" node receives the probability of correct "Speed Limit" detected as 70%. We see that the same final State S0 is selected, but with 70% probability. This result shows that with a sufficient probability from the PCNN, even when probability is less than the 100%, State S0 is correctly selected. This ensures that even when some uncertainty is observed, the car is still able to move.

– Test Case B3: as in Test Cases B1 ad B2, most of the safety conditions are met and the evidences provided to various nodes in a BN are the same, except for the "Valid Speed Limit" node. Here, the "Valid Speed Limit" node receives the probability of correct "Speed Limit" detected as 40% and therefore, we see that State S5 (Switch to ACC mode) is selected with 60% probability as the final output. Figure 2 shows that the state having the highest probability, i.e., State S5 (Switch to ACC mode), is selected. This represents the safest decision for this test case. Here, the output state selected changes because of low confidence in the validity of the speed limit (i.e. the evidence provided to the "Valid Speed Limit" node is below 50%, which is in this case, the acceptable safety threshold used in the BN). This test case shows that if blind trust is put into the "Speed Limit" detected from road sign boards, believing it to be always 100% accurate, then that is likely to lead to an unsafe output state. This was seen in the original implementation of the platooning case study, and would typically be the result if using advanced ML techniques like NN.

– Test Case B4: similar to the test cases above, most of the safety conditions are met, however, there is absolutely no confidence in the validity of the speed limit detected ("Validity of speed limit" is 0%) and therefore, final State S5 (Switch to ACC mode) is selected with 100% probability.

The Test Scenarios A and B show that while using BN, the Model and the Data Uncertainty (provided as normalised/probabilistic input to "Validity of speed limit" node) have a huge influence on the probability of the output state selected. The results show that our method of using a PCNN, to estimate both the Model and the Data Uncertainty, along with BN, enables us to make safe decisions. Unlike deterministic models, BN are capable of handling uncertainty in the input and therefore are an better choice for handling uncertainty generated from PCNN for making safe decisions.

6 Conclusion

In this paper, we have described how we can utilise the estimated uncertainty, arising from data and complex ML models, to improve safety in decision making. The proposed method allows designers of AV to improve the decision making process by integrating multiple sources of uncertainty. The efficacy of the proposed approach has been illustrated via an experimental analysis.

7 Acknowledgement

This research has received funding from Assuring Autonomy International Program (University of York) and the European Union's EU architecture Program for Research and Innovation Horizon 2020 under Grant Agreement No. 812.788

References

Burton, S.; Gauerhof, L.; and Heinzemann, C. 2017. Making the case for safety of machine learning in highly automated driving. In *International Conference on Computer Safety, Reliability, and Security*, 5–16. Springer.

Feng, D.; Rosenbaum, L.; and Dietmayer, K. 2018. Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 3266–3273. IEEE.

Furda, A.; and Vlacic, L. 2011. Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making. *IEEE Intelligent Transportation Systems Magazine* 3(1): 4–17.

Gal, Y.; and Ghahramani, Z. 2015. Bayesian convolutional neural networks with Bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158* .

Gauerhof, L.; Munk, P.; and Burton, S. 2018. Structuring validation targets of a machine learning function applied to automated driving. In *International Conference on Computer Safety, Reliability, and Security*, 45–58. Springer.

Graves, A. 2011. Practical variational inference for neural networks. In *Advances in neural information processing systems*, 2348–2356.

Hinton, G. E.; and van Camp, D. 1993. Keeping neural networks simple. In *International Conference on Artificial Neural Networks*, 11–18. Springer.

Houben, S.; Stallkamp, J.; Salmen, J.; Schlipsing, M.; and Igel, C. 2013. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *The 2013 international joint conference on neural networks (IJCNN)*, 1–8. IEEE.

Kabir, S.; Sorokos, I.; Aslansefat, K.; Papadopoulos, Y.; Gheraibia, Y.; Reich, J.; Saimler, M.; and Wei, R. 2019. A runtime safety analysis concept for open adaptive systems. In *International Symposium on Model-Based Safety and Assessment*, 332–346. Springer.

Katrakazas, C.; Quddus, M.; and Chen, W.-H. 2019. A new integrated collision risk assessment methodology for autonomous vehicles. *Accident Analysis & Prevention* 127: 61–79.

Kendall, A.; and Gal, Y. 2017. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, 5574–5584.

Koller, D.; and Friedman, N. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

Leung, H. W.; and Bovy, J. 2019. Deep learning of multi-element abundances from high-resolution spectroscopic data. *Monthly Notices of the Royal Astronomical Society* 483(3): 3255–3277.

Macfarlane, J.; and Stroila, M. 2016. Addressing the uncertainties in autonomous driving. *SIGSPATIAL Special* 8(2): 35–40.

Machin, M.; Guiochet, J.; Waeselynck, H.; Blanquart, J.-P.; Roy, M.; and Masson, L. 2016. Smof: A safety monitoring framework for autonomous systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48(5): 702–715.

McAllister, R.; Gal, Y.; Kendall, A.; Van Der Wilk, M.; Shah, A.; Cipolla, R.; and Weller, A. 2017. Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. International Joint Conferences on Artificial Intelligence, Inc.

Osband, I.; Blundell, C.; Pritzel, A.; and Van Roy, B. 2016. Deep exploration via bootstrapped DQN. In *Advances in neural information processing systems*, 4026–4034.

Papadoulis, A.; Quddus, M.; and Imprialou, M. 2019. Evaluating the safety impact of connected and autonomous vehicles on motorways. *Accident Analysis & Prevention* 124: 12–22.

Reich, J. 2016. *Systematic engineering of safe open adaptive systems shown for truck platooning*. Ph.D. thesis, M. Sc. thesis, Technical University of Kaiserslautern, Kaiserslautern, Germany.

Schwarting, W.; Alonso-Mora, J.; and Rus, D. 2018. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems* .

Stallkamp, J.; Schlipsing, M.; Salmen, J.; and Igel, C. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks* 32: 323–332.